Disease Discovery

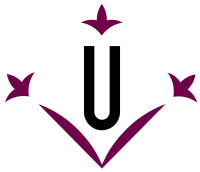
Sistema de predicció de malalties

Joel Badia Escolà

Directors

Francesc Solsona Tehas (Dept. Informàtica i Enginyeria Industrial)

Rui Carlos Vaqueiro de Castro Alves (Dept. Ciències mèdiques bàsiques)



Universitat de Lleida

EPS

Resum

L'objectiu d'aquest projecte consisteix en crear una eina per servir de referència a metges, per tal d'ajudar-los en els diagnòstics, sobretot de malalties més estranyes que són més difícils de diagnosticar, ja que no són tant conegudes. Estem interessats en realitzar una aplicació la qual donats uns símptomes ens doni les malalties més probables. El personal sanitari doncs podrà disposar d'una eina que l'ajudi a diagnosticar malalties segons els símptomes dels pacients.

L'eina que es presenta en aquest treball, Disease Discovery, disposa d'una base de dades omplerta des de fonts molt fiables i amb una quantitat elevada d'informació sobre malalties i símptomes. Aquesta base de dades s'entrena segons diferents classificadors. Les prediccions de les malalties es faran emprant un model generat per mitjà d'un entrenament realitzat amb un classificador sobre la base de dades. A més a més, s'ha realitzat una interfície web per ajudar als usuaris i fer que l'eina sigui més usable alhora que més còmoda de distribuir el programari entre el personal sanitari interessat.

S'ha posat a prova diversos classificadors, que ofereix la plataforma Weka, mesurant-n'he el seu rendiment. Mostrant els més favorables per l'ús final de l'aplicació, tant en temps de còmput com en la mida de memòria que utilitzen en l'entrenament i en la predicció.

Taula de continguts

Taula de Figures

1 Introducció

La mineria de dades és una tècnica d'intel·ligència artificial que es basa en convertir la informació en coneixement. La seva finalitat és convertir grans bases de dades en informació útil o coneixement, per tal de poder fer prediccions sobre aquesta informació basades en el coneixement adquirit en l'entrenament de sistemes, anomenats classificadors, que aprofiten com a font de coneixement les dades conegudes fins el moment.

Partint de la base que la mineria de dades ens permet fer prediccions sobre unes dades conegudes, neix la necessitat d'aplicar aquestes prediccions sobre molts àmbits que ho requereixen, com pot ser en els diagnòstics mèdics.

Fins ara aquest àmbit quedava reservat a la recerca d'informació en la bibliografia existent. El principal problema d'aquest sistema és que es tracta d'un sistema lent i que no s'actualitza gaire sovint. A més, és sensible a l'error humà i té el problema afegit que no hi són malalties rares i aquestes tenen que ser consultades en fonts separades.

Per altra banda, no hi ha cap sistema efectiu a nivell informàtic per determinar per mitjà d'un conjunt de símptomes d'entrada, un diagnòstic precís. Per exemple, una consulta en una base de dades per un conjunt de símptomes, et retornarà totes aquelles malalties que els contenen, en conjunt o individualment, depenent del criteri de la consulta. Per tant, no és un bon sistema per discriminar una malaltia d'entre les trobades en la consulta amb símptomes similars.

D'aquesta forma neix la necessitat de fer una aplicació que permeti oferir a la comunitat sanitària un sistema eficient, ràpid i segur de determinar un diagnòstic evitant els problemes explicats anteriorment. Així és com neix Disease Discovery, un sistema que aprofita la mineria de dades per cobrir aquesta necessitat, que fins ara no estava coberta.

Disease Discovery cobreix una sèrie de necessitats. La primera és la fiabilitat de la informació, ja que si aquesta no és fiable no seria útil en realitzar prediccions de malalties. Així que s'ha tingut que buscar fonts d'informació fiable a nivell mèdic, com per exemple, DiseaseCard, ORPHANET, RAMEDIS i NIH.

En segon lloc, cal oferir un sistema ràpid i eficient d'accés a les dades recopilades i relacionades entre elles. Aquesta informació en guarda en una BBDD relacional.

Finalment cal oferir un sistema robust i fiable en relació al soroll (símptomes no relacionats amb un diagnòstic), per assegurar la fiabilitat del sistema. En aquest apartat entra l'entrenament dels

diversos classificadors emprats en la mineria de dades que han d'assegurar aquesta resistència al soroll. Això s'ha fet generant “diagnòstics” amb diversos graus de soroll entorn a les malalties, simulant la injecció de falsos símptomes.

La inexistència d'una eina similar i la utilitat que aquesta pot suposar, ha motivat la realització d'una eina que cobreixi aquesta necessitat sobretot en l'aspecte de les malalties rares, al mateix temps que sigui accessible i de fàcil utilització dins d'aquest àmbit. Per aquest motiu l'eina oferirà una interfície web per facilitar la seva utilització i el seu accés.

2 Mètodes

2.1 Recol·lecció de dades

Primer de tot calia trobar una font fiable d'informació d'on obtenir les malalties i símptomes. Per fer-ho, s'ha fet un estudi previ de diversos llocs web amb una gran fiabilitat mèdica. Al mateix temps s'anava analitzant el seu funcionament, l'estructura estructura del web i el format de la informació que ens interessa per tal de realitzar un motor de cerca de les dades que contenen de forma automàtica.

A continuació es mostra el resultat de l'estudi realitzat de cada lloc web que ha estat catalogat com a font d'informació fiable.

2.1.1 DiseaseCard [http://www.diseasecard.org]

Anàlisi general del lloc web.

Els diferents camps rellevants dins del lloc web són els següents:

- References: Diferents referències així com una breu informació de la malaltia. (enllaç a omim).
- Pathology: Explicació de la malaltia (enllaç a orphanet).
- Drugs: Medicaments relacionats amb la malaltia (ellaç a PhramGKB).
- Clinical Trials: Proves mèdiques (enllaç a ClinicalTrials.gov).
- Genetic Tests and Labs: Dades i proves a nivell de laboratori (enllaç a NCBI).
- Pharmaco Research: Medicaments experimentals (enllaç a PhramGKB).
- Disorders and Mutations: És la mateixa informació que surt a References*.
- Polymorphism: Estudis sobre els gens (Enllaç a NCBI).
- Nucleoide Sequence: Seqüència nucleoide.
- Chromosomal Local: Informació sobre els cromosomes.
- Gene: Fitxa genètica de la malaltia (enllaç a HUGO Gene Nomenclature Comittee).
- Nomenclature: Nomenclatura de la malaltia.
- Enzyme: (Enllaç a swiss institute if Bioinformatics).
- Metabolic Pathway: Via metabòlica (enllaç a kegg).
- Functional Site: Lloc web de la malaltia
- Protein Structure: Estructura de la proteina (enllaç a Protein Data Bank).
- Protein Sequence: Seqüència de la proteina (enllaç a EMBL-EBI).
- Protein Domains: Dominis de la proteina (enllaç a EMBL-EBI).
- Gene Ontology: Ontologia genètica.

Aquesta web segueix el format més semblant al que tindria que tenir la versió definitiva d'aquest projecte. En la pàgina inicial es mostra una interfície molt simple, en la qual es pot fer cerques de

malalties, basades en diferents atributs.

Depenent de la informació recopilada de cada malaltia es mostra una barra de quantitat d'informació coneguda d'aquesta. Això pot donar una visió ràpida del coneixement que és té (almenys en el context d'aquest web) sobre la malaltia en qüestió.

2.1.2 RAMEDIS [https://www.imbio.de/stable/php/ramedis/htdocs/eng/index.php]

Anàlisi general del lloc web.

És un sistema relacionat amb les malalties rares del metabolisme. Entre d'altres camps, proporciona informació dels símptomes, descobriments del laboratori, teràpia i dades moleculars.

Un dels aspectes més interessants potser és el fet de que per cada diagnòstic, hi ha diversos casos d'estudi on es troba (de forma anònima) tot l'historial dels pacients.

La informació es troba distribuïda en pàgines (ordenades alfabèticament) de diagnòstics, on cada diagnòstic agrupa tots els casos registrats a la base de dades en un llistat. Per cada cas, es pot consultar l'històric de cada diagnòstic, on entre d'altres dades hi podem trobar els símptomes associats a cada malaltia.

2.1.3 NIH [http://www.nih.gov]

Anàlisi general del lloc web.

Aquesta pàgina té bastant informació, ja que té un llistat força gran de diagnòstics. Per cada malaltia, es pot veure que hi ha 5 tipus informació (camps):

- More detailed information: Amb informació sobre la malaltia i links externs amb informació més detallada d'aquesta.
- NLM Gateway: enllaç als recursos de la web National Library of Medicine.
- Scientfic Conferences: Informa de conferències a nivell de malalties en general.
- Suport groups: Organitzacions i empreses que ofereixen suport entorn a la malaltia.
- Clinical trials \& research: Enllaç a pàgines web sobre proves mèdiques i recerca.

De tots aquests camps l'únic rellevant per recopilar informació interessant és el primer, ja que els altres semblen donar la mateixa informació per totes les malalties. De totes formes el primer et porta a web externes amb més informació i per tant no dóna una forma estandarditzada que permeti una manera efectiva de tractar la informació.

Per tant és difícil obtenir informació de forma automàtica en aquest lloc web.

2.1.4 ORPHANET [http://www.orpha.net/consor/cgi-bin/index.php]

Anàlisi general del lloc web.

Aquesta web té volum d'informació molt gran, a més a més, mitjançant la pàgina www.orphadata.org, que s'ofereix al lloc web per poder treballar còmodament amb la informació de la que disposa, proporciona un xml d'on extreure la informació de forma ràpida i efectiva, on es relacionen malalties amb els seus símptomes associats, fet que facilita enormement la recol·lecció de la informació.

2.2 Model relacional

El model relacional és l'encarregat de permetre, entre altres coses, una consulta ràpida de la informació. Ja que avaluant les necessitats de la plataforma s'estableixen relacions entre la informació a tractar i aquesta és emmagatzemada de forma eficient per tal d'optimitzar el tractament d'aquestes dades.

El model relacional és el que es pot veure, gràficament en un diagrama UML, de la Figura 1. Està format per 3 taules, una de “symptoms” i una altra de “diseases” on pot haver-hi una relació de 0 a n malalties per “symptom” o a la inversa. Al final les relacions s'emmagatzemen a la taula diagnosis, que té com a claus foranes, disease_id i symptom_id.

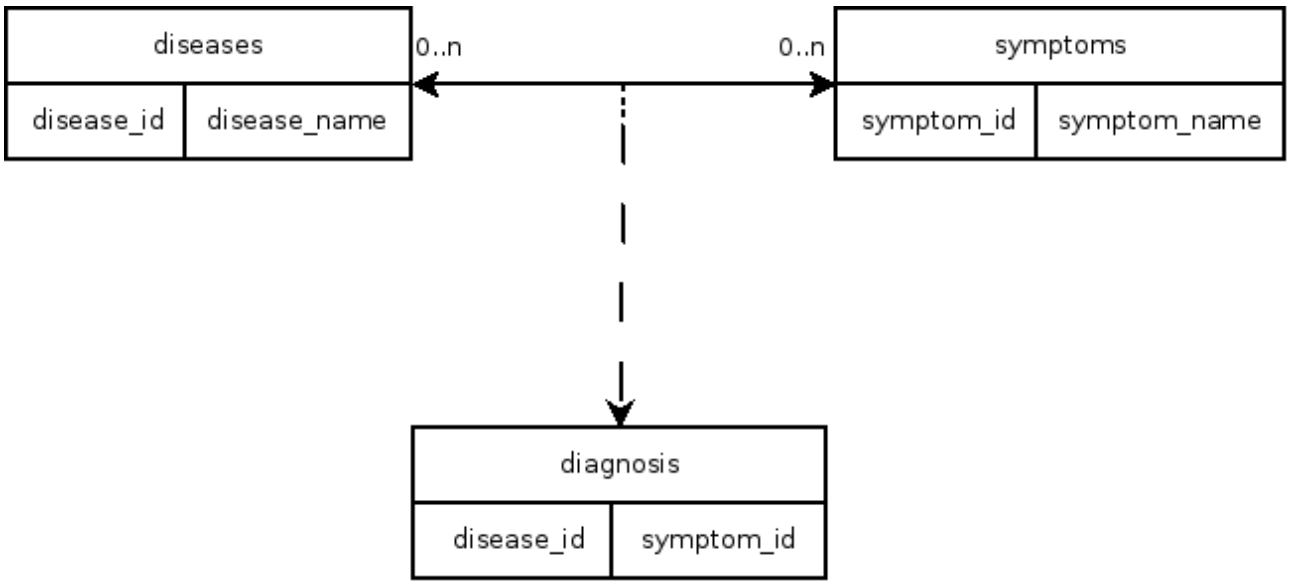


Figura 1: Model relacional de la base de dades

2.3 Funcionament de la plataforma

El funcionament de la plataforma es basa en diversos mòduls, que treballen conjuntament per suplir les diverses necessitats d'aquesta. A continuació s'esmentaran els diversos mòduls que formen el conjunt de l'aplicació.

2.3.1 Mòdul de captació de dades

Aquest mòdul és l'encarregat de mantenir la base de dades de l'aplicació actualitzada, per fer-ho té que comptar amb diversos mòduls que s'encarreguin d'extreure dades de tots els llocs esmentats anteriorment, que contenen informació útil, en l'apartat "Recol·lecció de dades".

Per fer-ho fa falta mètodes per poder obrir connexions amb el protocol Http i connexió a bases de dades MySQL (el sistema de gestió de base de dades que s'ha escollit). Degut a què gran part de l'aplicació estarà feta amb Java per tenir una certa hegemonia i també pel fet que té un SDK (JDK) molt complet, és la tecnologia que s'emprarà en la implementació d'aquests mòduls.

A continuació s'esmentaran les tecnologies emprades en aquest mòdul.

Java

"El Java és un llenguatge de programació dissenyat el 1990 per James Gosling amb altres companys de Sun Microsystems a partir de C++. Des del seu naixement fou pensat com un llenguatge orientat a objectes. Entre el 13 de novembre de 2006 i el maig del 2007 Sun va alliberar parts de Java com a programari lliure de codi obert amb llicència GPL. És un dels llenguatges de programació més utilitzats, i s'utilitza tant per aplicacions web com per aplicacions d'escriptori.

El Java és un llenguatge interpretat i, per tant, pot semblar lent en comparació amb altres

Figura 2: Logotip java

llenguatges, però ofereix un índex de re-utilització de codi molt elevat, sent possible trobar moltes

llibreries lliures de Java."[1]

S'ha escollit per desenvolupar l'aplicació degut a les seves característiques, principalment per ser un llenguatge multiplataforma, pel complet i potent SDK, i pel conjunt d'eines de les que disposa.

MySQL

"MySQL és un sistema de gestió de bases de dades relacional (anglès RDBMS - Relational DataBase Management System) multi-fil (multithread) i multiusuari, que usa el llenguatge SQL (Structured Query Language).

MySQL és molt utilitzat en aplicacions web, com Drupal o phpBB, en plataformes (Linux/Windows-Apatxe-MySQL-PHP/Perl/Python), i per eines de seguiment d'errors com Bugzilla. La seva popularitat com a aplicació web està molt lligada a PHP, que sovint apareix en combinació amb MySQL. MySQL és una base de dades molt ràpida en la lectura quan utilitza el motor no transaccional MyISAM, però pot provocar problemes d'integritat en entorns d'alta concurrència en la modificació. En aplicacions web hi ha baixa concurrència en la modificació de dades i en canvi l'entorn és intensiu en lectura de dades, la qual cosa fa a MySQL ideal per a aquest tipus d'aplicacions."[2]

Figura 3: Logotip MySQL

Degut a la seva popularitat, que és de codi obert i a la seva elevada velocitat de consulta el fa el sistema perfecte.

En la Figura 1, es pot veure com el model relacional de la nostra base de dades es força senzill però molt eficient pel tipus de consultes que s'hi volen fer.

2.3.2 Mòdul d'entrenament

En aquest mòdul es tenen que generar fitxers d'entrenament que pugui entendre el framework de

1 Descripció extreta de la Viquipèdia: http://ca.wikipedia.org/wiki/Java_%28llenguatge_de_programaci%C3%B3%29
2 Descripció extreta de la Viquipèdia: http://ca.wikipedia.org/wiki/Mysql

mineria de dades Weka (basat en Java), amb el format natiu del framework anomenat “arff” (Attribute-Relation File Format). A més a més, ha de llençar els entrenaments amb diversos classificadors triats. Més endavant es descriurà els diversos classificadors que s'han emprat i quines dades i decisions s'ha pres respecte ells. En aquest apartat es descriurà Weka i el seu funcionament, així com el format dels fitxers “arff” utilitzat.

Per poder automatitzar els entrenaments, s'ha utilitzat shell scripts programats en bash.

Attribute-Relation File Format (arff)

El format Attribute-Relation File Format (arff)[3] és el format de dades amb que Weka treballa nativament, n'accepta d'altres, però és més fiable l'ús d'aquest format.

Els fitxers amb format arff utilitzen caràcters amb la codificació ASCII. Cada fitxer conté un llistat d'instàncies que comparteixen un mateix conjunt d'atributs. Està dividit en dos zones principals:

1. La secció de capçalera (Header Section): Aquesta secció defineix el format de les dades que conté el fitxer. @RELATION <nom_relació>, té que encapçalar el fitxer i ofereix el nom a la relació, és a dir, el de tot el conjunt de dades que conté el fitxer arff. @ATTRIBUTE <nom_atribut> <tipus_dades> aquesta secció pot estar formada per un nombre variable d'atributs, i aquests poden ser de diversos tipus. L'ordre en que es posen s'haurà de respectar en el moment de crear les instàncies, ja que és la forma que Weka ho pugui interpretar correctament. Els tipus d'atributs que hi ha, poden ser:
    - Numeric: poden ser enters o reals.
    - Nominal: contenen un llistat de noms, com per exemple {nom1, nom2, etc.}
    - String: contenen un valor arbitrari de caràcters, és a dir, que poden tenir com a valor qualsevol cadena de caràcters.
    - Date: conté la data.
2. La secció de dades (@DATA): En aquesta secció es defineixen totes les instàncies que contindrà el fitxer, seguint estrictament l'ordre de com s'ha definit els atributs. Els diversos valors que formen una instància tenen que estar separats per comes i cada instància al mateix temps es té que separar de les altres per canvis de línia.

3 Més informació sobre el format a http://www.cs.waikato.ac.nz/ml/weka/arff.html

Waikato Environment for Knowledge Analysis (WEKA)

Aquest és l'entorn/framework des del que es farà l'entrenament de les dades, per posteriorment poder fer prediccions basades en l'aprenentatge generat sobre aquestes. Tant en les proves, entrenaments, obtenint estadístiques, etc. pot mostrar els resultats tant gràficament com per línia de comandes.. Al mateix temps que és un framework, ja que ofereix una api molt completa, on es pot integrar tot el sistema de Weka a les aplicacions Java.

Figura 4: Logotip Weka

Weka disposa d'una GUI (Graphical User Interface) i d'un CLI (command line interface). Per molt que la interfície gràfica tingui la seua utilitat, en un entorn productiu, es té que poder automatitzar tot el sistema d'entrenament, i la interfície gràfica no permet aquesta possibilitat, ja que es té que llençar cada entrenament un per un. Per altra banda té un altre inconvenient i és que desar tant els models com els resultats s'ha de fer manualment en dos passos diferents.

Per aquests motius la GUI no és la solució i cal emprar la CLI (command line interface), per poder automatitzar tots aquests processos. Per poder emprar la CLI des d'un interpret de comandes (no des de l'opció que ofereix la GUI de CLI que té les mateixes limitacions anteriorment esmentades), hi ha dues possibilitats, la primera és carregar al classpath de la màquina virtual de java el paquet jar de weka i la segona descomprimir el jar de weka, en els dos casos per emprar un classificador caldrà emprar el "Qualified Package Name"[4] de weka, és a dir, weka.classifiers.nom_classificador.

2.3.3 Mòdul d'aplicació d'usuari

Aquest mòdul és l'encarregat d'interactuar amb l'usuari final de l'aplicació, que com s'ha comentat hi interactuarà mitjançant una interfície web. Degut a aquesta arquitectura i sabent que Weka funciona sobre Java, facilitarà molt la feina l'ús del contenidor de servlets Apache Tomcat, ja que permetrà incloure les llibreries de Weka sense cap problema. A més a més al tenir un servidor http integrat també se'l considera un servidor web independent.

L'ús d'una interfície web engloba una sèrie de tecnologies que es descriuran a continuació.

4 Especificació de Java sobre QPN: http://docs.oracle.com/javase/specs/jls/se5.0/html/names.html#6.5.3

Tomcat

"Apache Tomcat (abans sota el projecte Apache Jakarta) és un contenidor de servlets desenvolupat a l'Apache Software Foundation. Tomcat implementa les especificacions de servlet i de Java Server Pages (JSP) de Sun Microsystems, proporcionant un entorn per al codi Java a executar en cooperació amb un servidor web. Aquest afegeix eines per a la configuració i el manteniment, però també pot ser configurat editant els fitxers de configuració que normalment són en format XML. Tomcat inclou el seu propi servidor HTTP, per això també se'l considera un servidor web independent."[5].

Figura 5: Logotip Tomcat

Principalment ofereix l'entorn perfecte per fer aplicacions web basades en Java, de fet és una de les millors implementacions que hi ha de les especificacions de servlets i JSP, al mateix temps que la seua llicència Apache, el converteixen en el contenidor de servlets més famós, que empren tant solucions privatives com FOSS[6] per a completar la seva implementació de JEE[7]. Fet que facilita trobar-hi documentació o ajuda al respecte.

Java Server Pages

"JavaServer Pages és una tecnologia que permet als desenvolupadors de pàgines web, generar respostes dinàmicament a peticions HTTP. La tecnologia permet que codi Java i certes accions predefinides siguin incrustades en un context estàtic.

La sintaxi de JSP incorpora tags XML addicionals, anomenats accions de JSP, per ser usats per invocar altres funcions. Addicionalment, la tecnologia permet la creació de llibreries d'etiquetes que actuen com extensions de l'estàndard d'etiquetes HTML o XML. Les llibreries d'etiquetes aporten una forma multiplataforma d'ampliar les capacitats d'un servidor web.

Els JSPs són compilats en Servlets per un compilador JSP. Aquest pot generar un servlet o generar

5 Descripció extreta de la Viquipèdia: http://ca.wikipedia.org/wiki/Apache_Tomcat
6 Free Open Source Software
7 Java Enterprise Edition

bytecode (codi binari intermedi que interpreta la màquina virtual de Java) directament."[8].

Gràcies a la tecnologia JSP, tal com es diu a la descripció, ens permet treballar de forma dinàmica en un context estàtic. Això ve donat pel compilador de JSP Jasper, que és capaç de transformar el codi JSP que es pot barrejar amb l'HTML com passa en el PHP, per exemple, a bytecode o servlets, d'aquesta forma s'aconsegueix una facilitat de treball (no cal compilar el codi del servlet cada cop, se n'encarrega Jasper automàticament cada cop que detecta que un fitxer ha canviat) així ofereix l'experiència de treballar amb un llenguatge d'scripting fet que agilitza moltíssim el desenvolupament.

Al mateix temps, cal tenir cura del bon disseny. Per exemple dins de l'HTML el codi inclòs té que ser mínim ja que sinó es barregen funcionalitats i complica el manteniment, per aquest motiu excepte tasques molt trivials, la resta es posaran en paquets o classes, dins els directoris de l'aplicació WEB-INF/{lib,classes}, que després es cridaran dins del fitxer jsp.

jQuery

"jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript."[9].

Si s'ha d'implementar una aplicació web, s'ha de fer ús de Javascript per oferir una bona interacció amb l'usuari, natural, i eficient (entenent eficiència com la rapidesa de resposta i la reducció de trafic, en volum, entre servidor i client).

A més a més, ofereix un sistema de baix pes, que s'adapta a les possibles diferències entre motors JavaScript, i ofereix una forma molt bona de treballar amb HTML sense barrejar-se amb el JavaScript, gràcies al sistema de "selectors" i "events".

8 Descripció extreta de la Viquipèdia: http://ca.wikipedia.org/wiki/JSP
9 Descripció de jQuery del lloc web oficial: http://jquery.com/

3 Resultats

En aquest apartat, s'exposaran els resultats obtinguts,en l'avaluació del bon rendiment i bon funcionament de l'aplicació. S'avaluaran també diferents alternatives de disseny de “Disease Discovery”.

3.1 Recol·lecció de dades

Anteriorment s'ha analitzat la informació útil dels llocs web sobre els que s'hi té que extreure informació. Ara ens centrarem en els que disposen d'una font d'informació tractable i de qualitat (considerant informació única respecte als altres llocs i abundant), i avaluarem els sistemes per poder extreure'n la informació.

3.1.1 RAMEDIS [https://www.imbio.de/stable/php/ramedis/htdocs/eng/index.php]

Anàlisi tècnic per la recol·lecció de dades.

Primerament es partirà del punt on es fa la cerca al lloc web: https://www.imbio.de/stable/php/ramedis/htdocs/eng/index.php

El primer que es troba és tot el conjunt de diagnòstics ordenats alfabèticament i mostrats en diverses pàgines.

A continuació s'obtenen tots els enllaços a les pàgines de diagnòstics de cada malaltia, per fer-ho cal analitzar el codi html. Es pot veure que totes les pàgines amb enllaços a diagnòstics segueixen el patró següent: href="search_diagnosis_overview.php?suche=\%\&shown=. Per tant, tan sols es té que agafar el valor del camp href per tenir l'enllaç relatiu a la següent pàgina. Per formar la url correctament tenim que concatenar el valor que s'ha obtingut de href a l'adreça https://www.imbio.de/stable/php/ramedis/htdocs/searchv2/.

Per altra banda, mentre es va saltant de pàgina en pàgina és una bona idea anar guardant els enllaços als diversos diagnòstics que contenen aquestes, ja que així es redueix la càrrega excessiva al servidor i es pot evitar des de caigudes fins a expulsions degut a l'excés de peticions. Per fer-ho es té que obtenir el contingut de l'atribut href en les línies html que compleixin el patró <a href="search_diagnosis_ok.php?id=.

Un cop emmagatzemats els enllaços, s'accedeix a tots els diagnòstics que conté RAMEDIS, s'hi accedeix mitjançant la url amb el format https://www.imbio.de/stable/php/ramedis/htdocs/searchv2/ + search_diagnosis_ok.php?id=num_id.

Aquests enllaços que els anirem formant un a un, porten a una pàgina amb diversos diagnòstics associats a una mateixa malaltia. Per tant, és la situació perfecta per poder contrastar símptomes dins una mateixa malaltia. Per aconseguir la url lligada a cada diagnòstic en particular, cal afegir a la url base https://www.imbio.de/stable/php/ramedis/htdocs/ el valor de l'atribut “a” dins l'”html” d'aquelles línies que segueixin el patró <A HREF="../dm-guest/case_main.php?patID=, per tant a la url base s'hi afegirà una url relativa amb la forma ../dm-guest/case_main.php?patID=id i així ja es pot arribar fins un diagnòstic concret.

Un cop s'està en un diagnòstic cal obtenir les malalties relacionades a aquest, per fer-ho, es pot fer una petició al servidor per a que executi l'”script “ case_symptoms.php. És a dir, per cada diagnòstic es construirà la url https://www.imbio.de/stable/php/ramedis/htdocs/dm-guest/case_symptoms.php. Es té que tenir en compte que la url és la mateixa per tots els diagnòstics, això és degut a què l'”script” fa ús de galetes que emmagatzemen al costat client des de quin diagnòstic es fa la petició, per tant a la font de recol·lecció d'informació se li té que indicar que emmagatzemi les galetes, ja que per defecte en una petició http no és fa i simplement són ignorades.

Repetint el procés per tots els diagnòstics s'obté tota la font de coneixement que emmagatzema RAMEDIS.

3.1.2 ORPHANET [http://www.orpha.net/consor/cgi-bin/index.php]

Anàlisi tècnic per la recol·lecció de dades.

Aquest anàlisi es fa sobre un fitxer xml que ens proporciona la pàgina www.orphadata.org per treballar amb la informació que conté ORPHANET. L'adreça del fitxer és http://www.orphadata.org/data/xml/en_product4.xml.

Tot llenguatge de programació modern conté sistemes molt eficients i robustos per tractar fitxers de tipus xml. Per fer-ho només cal conèixer quins “tags” té el fitxer i quina jerarquia segueixen entre ells. L'anàlisi a continuació ho mostra.

- [Tag Disease] En aquesta part s'està treballant amb una malaltia determinada.
- [Tag Name] Proporciona el nom de la malaltia.
- [Tag DiseaseSign] Indica que aquest símptoma està associat a la malaltia actual.
- [Tag Sign] Sabem que dins del símptoma actual, aquest té una sub-etiqueta amb el nom Name, que ens proporciona el nom del símptoma, així podem relacionar el símptoma amb la

malaltia.

- [Tag SignFreq] Ens proporciona la freqüència del símptoma actual.

3.2 Classificadors (Classifiers)

Els classificadors (classifiers), es tracten d'algorismes d'aprenentatge en mineria de dades, que empren un conjunt de dades inicials, amb que basen el seu coneixement, per tal de fer prediccions.

Tot classificador en mineria de dades, aquest consta de dos fases, la fase d'aprenentatge i la de classificació. En la d'aprenentatge es crea una estructura de dades anomenada model que es forma d'acord amb unes dades d'entrada i les condicions d'aprenentatge de cada classificador. El conjunt d'aquestes dades d'entrada s'anomena “dataset”.

Aquests juguen el paper principal, dins de la mineria de dades, així com en l'aplicació, ja que es tracten de la columna vertebral d'aquesta.

Entre d'altres classificadors que ofereix Weka, en aquest projecte s'han utilitzat els següents:

- Random Tree
- Random Forest
- SMO (Support Vector machines)
- HNB (Hidden Naive Bayes)

Els entrenaments preparats per tots aquests classificadors estan basats en generar fitxers anomenats “dataset” amb un conjunt de dades, que s'empraran per fer l'aprenentatge de tota la base de dades de malalties i símptomes que s'ha recol·lectat, el funcionament del format d'aquests fitxers està explicat en l'apartat de Attribute-Relation File Format (arff), explicat anteriorment. En els entrenaments no és massa important el temps o la memòria consumida (dins dels límits del que és físicament i temporalment acceptable), ja que un cop generats els models de per dur a terme les prediccions, tant el temps com l'ús de memòria son menors respecte a l'entrenament.

En l'entrenament s'ha generat 4 dataset per classificador. Les entrades en aquests dataset simulen pacients, on s'indica la malaltia i els símptomes associats. A continuació es mostra un exemple d'entrada:

14562,y,n,y,n,n,n,n,n,y,...,n

On el primer nombre és un identificador de la malaltia i els {y,n} fan referència a atributs

símptoma, seguint l'ordre establert en la capçalera (*y*: yes, vol dir que té el símptoma i *n*: no, vol dir que no el té).

Els quatre dataset generats es diferencien en el tipus de soroll que se'ls hi aplicat. Per cada malaltia es generen 20 pacients. Depenent del tipus d'entrenament en tindrem sense soroll, amb soroll intern (dins dels símptomes relacionats amb la malaltia), amb soroll extern (probabilitat d'afegir símptomes externs), i soroll extern i intern barrejats. El soroll es genera en tots els casos amb un factor de probabilitat del 10%.

L'objectiu de fer aquests 4 entrenaments és per dotar-lo de més robustesa, al mateix temps sense afegir massa soroll, per no produir massa col·lisions entre les diverses malalties i generant un nombre de pacients suficient com per aconseguir el mínim de xocs entre les diferents malalties. Tant el nivell de soroll com de pacients s'ha decidit després de diverses proves d'ajustat del sistema.

Finalment de cara a la predicció final, es llançaran els 4 testos, per cada classificador i el/els resultat més predominant serà el que es mostrarà.

Els resultats dels entrenaments ofereixen unes estadístiques que permeten avaluar la qualitat de la classificació:

1. Correctly Classified Instances: Valor absolut i percentatge, d'instàncies correctament classificades. Aquest percentatge també s'anomena precisió.
2. Incorrectly Classified Instances: Valor absolut i percentatge, d'instàncies incorrectament classificades.
3. Kappa statistic: és una mesura de probabilitat corregida entre les classificacions i les classes reals. Un valor superior a zero ens està dient que les classificacions són millors que una predicció feta a l'atzar.
4. Valors d'error: Negligibles en el context en que s'ha treballat, ja que s'empren per prediccions numèriques i no és el nostre cas.
5. Total Number of Instances: Nombre total d'instàncies en el “dataset” emprat en l'entrenament.

A continuació s'analitzarà com s'ha fet l'entrenament en cadascun, com funcionen i com fer prediccions emprant Weka.

3.2.1 Random Tree (Arbre de decisió)

Un arbre de decisió no deixa de ser un conjunt de normes organitzades en una estructura jeràrquica, ja que per arribar una decisió es té que començar al node arrel i anar seguint tot l'arbre fins arribar a un node fulla (mai es tornarà enrere).

Per tal de fer prediccions, un arbre de decisió, porta a terme diversos tests, mentre es va recorrent. En aquest punt entra en joc els diferents tipus d'elements que el formen:

- Node intern: conté un test sobre algun valor de les propietats.
- Node probabilístic: indica algun tipus d'esdeveniment aleatori, basat en el problema.
- Node fulla: valor que retornarà l'arbre.
- Branques: elements de connexió entre els diferents nodes.

Un cop fet l'entrenament, es té un arbre que té un coneixement basat en les dades conegudes en el moment d'aprenentatge. Mitjançant aquest arbre, donades les dades d'entrada, es poden fer prediccions (sempre tenint en compte que estaran condicionades sobre les dades utilitzades en el moment de l'entrenament).

Una predicció consisteix en anar llegint les dades d'entrada i seguir els camins que marca l'arbre d'acord a les condicions o probabilitats especificades en cada node de l'arbre, fins arribar en un node fulla.

Weka disposa d'utilitats d'aprenentatge i classificació, accessibles des de la CLI (Command Line Interface). Per llençar un entrenament amb un Random Tree a Weka mitjançant la CLI, cal executar l' ordre següent:

- *java -Xmx75000m weka.classifiers.trees.RandomTree -t data/ft-nonoise-p20-o0-i0.arff -c 1 -d results/models/$ALG-nonoise.model > results/$ALG-nonoise.txt*

On l'argument -Xmx expecifica el “heap size” de la màquina virtual de Java, aquí es pot observar (al tractar-se d'un valor real) la quantitat impressionant que fa falta tenir de memòria RAM per poder fer l'entrenament, degut al gran dataset generat amb les dades de tota la base de dades de diagnòstics. L'opció -t indica on en troba el dataset amb les dades d'entrenament. L'argument -c especifica quin és l'index dins de tots els atributs (definits a l'apartat de capçalera) sobre el que es farà l'entrenament (aquests índex, per lògica, tenen que ser els mateixos en l'entrenament i predicció). En el cas de l'opció -d s'especifica on es vol guardar el model generat (és a dir l'arbre de decisió, per després utilitzar-lo). Finalment la redirecció de la sortida estàndard serveix per guardar

en un fitxer tota la informació que s'ha generat durant l'entrenament (confusion matrix, error de classificació, etc.).

Un cop s'ha obtingut el model generat en l'entrenament, ja es pot fer la predicció. La forma de fer-ho amb Weka és:

- *java -Xmx10000m weka.classifiers.trees.RandomTree -T data/test_dataset.arff -c 1 -l results/models/$ALG-nonoise.model -p 0*

Algunes de les opcions ja han set explicades amb anterioritat, així que només s'explicaran aquelles que són noves. Primer l'opció -T especifica on es troba el dataset per fer la predicció. En segon lloc -l especifica on hi ha el model generat amb anterioritat. I finalment -p indica quins atributs sortiran al costat de la predicció, s'ha posat un 0, el qual vol dir que no en sortirà cap. Per interpretar la sortida estàndard de la predicció cal saber quin és el format:

- *<test_instance_index><actual_class_index>:<actual_class_val><pred_class_index>:<pred_class_val> [+|] <prob_of_pred_class_val>*

Aquest és el format de la sortida, els valors importants són primer el actual_class_val, que ens indica quin valor tenia, després el pred_class_val, quin és el valor després de la predicció i finalment prob_of_pred_class_val per saber en quina precisió s'ha fet la predicció.

Resultats

A continuació es veuran i s'analitzaran els resultats rellevants de l'entrenament realitzat amb tots els datasets.

- Sense soroll:

Correctly Classified Instances	53780	100%
Incorrectly Classified Instances	0	0%
Kappa statistic	1	
Total Number of Instances	53780	

- Soroll intern:

Correctly Classified Instances	53780	100%
Incorrectly Classified Instances	0	0%

Kappa statistic	1	
Total Number of Instances	53780	

- Soroll extern:

Correctly Classified Instances	53780	100%
Incorrectly Classified Instances	0	0%
Kappa statistic	1	
Total Number of Instances	53780	

- Soroll intern i extern:

Correctly Classified Instances	53780	100%
Incorrectly Classified Instances	0	0%
Kappa statistic	1	
Total Number of Instances	53780	

El més important de totes aquestes dades, és el percentatge d'instàncies classificades correctament i les que no. Segons aquests resultats, no hi ha cap mena de creuament entre les dades. Aquest fet és ideal, ja que assegura que en el cas dels Random Tree hi haurà prediccions 100% fiables, respecte a la informació dels dataset generats per l'aprenentatge. Per altra banda degut a la simplicitat de l'arbre, els temps d'entrenament i predicció són força bons, ja que ronden entorn als 30 min i 1 min respectivament.

3.2.2 Random Forest

Sabent el que és un arbre de decisió, un Random Forest es pot definir com un conjunt d'arbres de decisió. Les dades d'entrada es proven en cadascun dels arbres, i retorna l'opció més “votada” de totes les prediccions.

Mitjançant Weka la forma de treballar amb Random Forest és la següent:

- *java -Xmx75000m weka.classifiers.trees.RandomForest -t data/ft-nonoise-p20-o0-i0.arff -c 1 -d results/models/$ALG-nonoise.model -I 5 > results/$ALG-nonoise.txt*

Els avantatges són evidents, respecte als arbres de decisions, si tenim en compte que es poden contemplar molts més escenaris.

Els paràmetres definits en general són els mateixos que en el cas del Random Tree, ja que Weka ofereix una interfície estandarditzada, dins dels possibles, per tots els classificadors. L'únic paràmetre diferent és -I que especifica el nombre d'instàncies (arbres) a generar, només se n'especifica 5 pel simple motiu de les limitacions en quant a RAM, ja que més no n'és capaç de suportar el sistema físicament, ja que supera el límit de memòria disponible.

En l'àmbit de la predicció en Weka es llençaria l'opció següent:

- *java -Xmx10000m weka.classifiers.trees.RandomForest -T data/test_dataset.arff -c 1 -l results/models/$ALG-nonoise.model -p 0*

No varia res, respecte a l'anterior apartat, excepte el classificador implicat.

Resultats

A continuació es comentaran els resultats obtinguts en l'entrenament.

- Sense soroll:

Correctly Classified Instances	53688	99.8289 %
Incorrectly Classified Instances	92	0.1711 %
Kappa statistic	0.9983	
Total Number of Instances	53780	

- Soroll intern:

Correctly Classified Instances	53182	98.8881 %
Incorrectly Classified Instances	598	1.1119 %
Kappa statistic	0.9889	
Total Number of Instances	53780	

- Soroll extern:

Correctly Classified Instances	53182	98.8881 %

Incorrectly Classified Instances	598	1.1119 %
Kappa statistic	0.9889	
Total Number of Instances	53780	

- Soroll intern i extern:

Correctly Classified Instances	53150	98.8286 %
Incorrectly Classified Instances	630	1.1714 %
Kappa statistic	0.9883	
Total Number of Instances	53780	

Després d'obtenir els resultats es pot veure que a diferència d'abans amb els Random Tree, aquests classificadors tenen algun error d'instàncies mal classificades, és possible, ja que s'ha generat 5 arbres per cada entrenament, i és possible que hi hagi més probabilitat d'error degut al factor aleatori amb que es creen els arbres. De totes maneres és un error molt petit i és pot acceptar per fer prediccions com a bo.

Per altra banda, l'aplicació real d'aquest sistema, al tenir que provar els diversos arbres, i al no estar implementat de forma que aprofiti la computació paral·lela (ja que podria testejar diversos arbres simultàniament), entre altres coses per la impossibilitat de fer-ho en Java 6. El temps de predicció és exageradament lent i no és factible per ser emprat en l'aplicació final, ja que pot tardar entorn a 10 min en fer una predicció. Per altra banda el temps d'entrenament és totalment acceptable ja que ronda l'hora.

3.2.3 SMO (Support Vector Machines)

La primera versió de SVM es basava en fer prediccions binàries, ja que la idea principal de l'algorisme es basa en crear un vector de suport, que defineix una separació entre totes les dades d'un dataset. Les instàncies només poden ser classificades com una de les dos categories, segons si queden en un costat o altre del vector de suport. La posició on pertanyen les instàncies queda determinada a través de la funció kernel.

Tot i això actualment hi ha noves actualitzacions que entre d'altres, ofereixen la possibilitat de treballar en més d'una categoria, concretament a aquesta variació de SVM se l'anomena de

multiclasse i és la que implementa Weka amb SMO conjuntament amb una sèrie d'optimitzacions (anomenades Sequential Minimal Optimization) que s'han anat afegint des de les primeres versions del classificador. Si el nombre de categories, en el nostre cas nombre de malalties, és superior a dos, hi ha dos sistemes per atacar el problema:

1. Cada categoria és dividida en altres i totes són combinades.
2. Es construeixen n(n-1)/2 models on n és el nombre de categories.

El sistema de les funcions kernel de cara a fer les prediccions és molt eficient (en termes de velocitat de predicció), ja que com en el cas de la funció de distribució d'un hash, aquesta ràpidament pot determinar en quin punt va destinada una instància i per tant aconseguir una velocitat de predicció molt bona. A més a més, aquests sistemes acostumen a fer prediccions més precises que els Random Trees o Forest.

El principal problema i que no ha permès fer possible l'ús dels d'aquest classificador és la falta de RAM, ja que amb 98Gb no n'hi ha hagut suficient en l'entrenament. Fet que de moment l'ha situat com a no factible per l'ús de l'aplicació actual, conjuntament amb el temps de classificació (més d'una setmana abans de superar el límit de memòria disponible).

3.2.4 HNB (Bayesian Networks)

Formalment una xarxa bayesiana es tracta d'un graf acíclic no dirigit, que vol dir que no hi ha camins cíclics, és a dir que no pots retornar per un camí a l'inici del punt on has començat a seguir el graf, i dirigit per què com el seu nom indica els arcs del graf només tenen un sentit.

En aquest tipus de grafs, els nodes són variables, mentre que els arcs que uneixen els nodes codifiquen dependències condicionals entre els nodes, és a dir allò que els relaciona. Per generar el model es basa totalment en càlculs probabilístics.

Així que el càlcul de prediccions d'aquest sistema és totalment probabilista.

Degut a que es poden crear xarxes molt complexes amb molts nodes, sobretot en grans dataset, aquestes requereixen d'un gran esforç computacional i d'una gran quantitat de memòria, fet pel qual no són factibles a nivell pràctic. Els entrenaments que s'han dut a terme amb aquest classificador ho ha corroborat ja que s'ha sobrepassat el límit físic del que es disposava, 98Gb de RAM.

3.3 Aplicació Disease Discovery (GUI)

En aquest apartat s'explicarà el funcionament de l'aplicació, així com es mostraran algunes imatges

que facilitin la comprensió d'aquest funcionament. L'aplicació ofereix tres funcionalitats bàsiques.

La primera funció és la predicció de malalties, per fer-ho, cal introduir símptomes al quadre de l'esquerra que ofereix funció d'autocompleció i seleccionar aquells que vols introduir. Un cop es tenen tots els símptomes desitjats seleccionats, cal clicar al botó “Diagnòstic”, veure la Figura 6 i s'obtindrà el resultat com es veu en la Figura 7. Aquest resultat és correcte, ja que s'ha comprovat fent una consulta a la base de dades i s'ha corroborat que és la malaltia que està relacionada amb aquests símptomes. És a dir, s'ha buscat tots aquells símptomes relacionats amb l'Alexander Disease i s'ha comprovat que efectivament el resultat de la predicció era correcte.

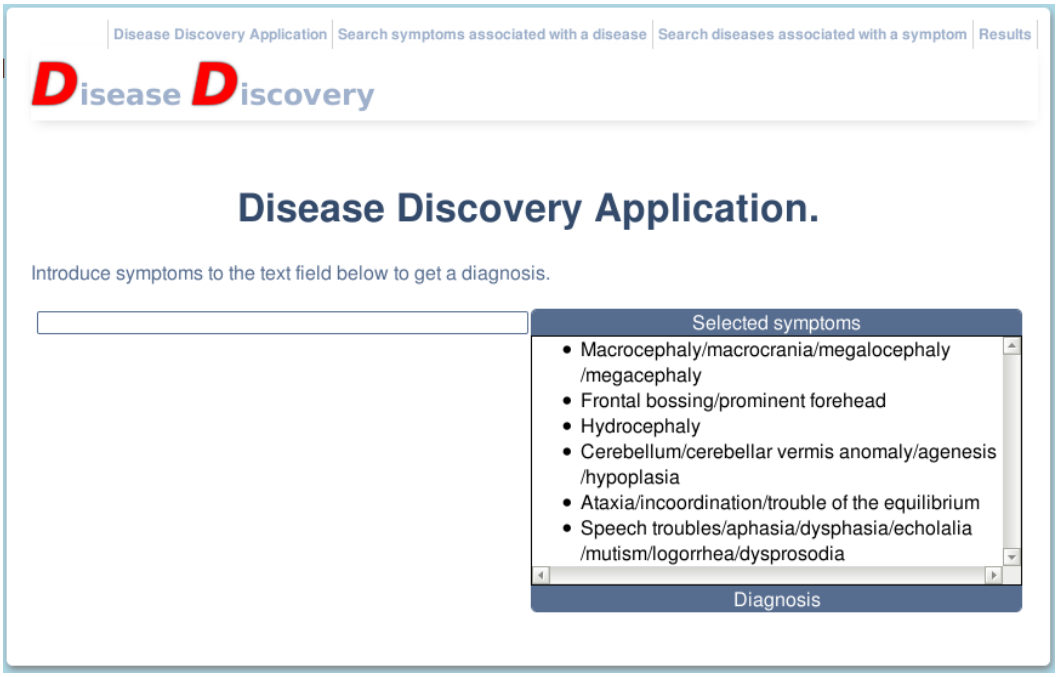


Figura 6: Seleccionant símptomes per fer una predicció

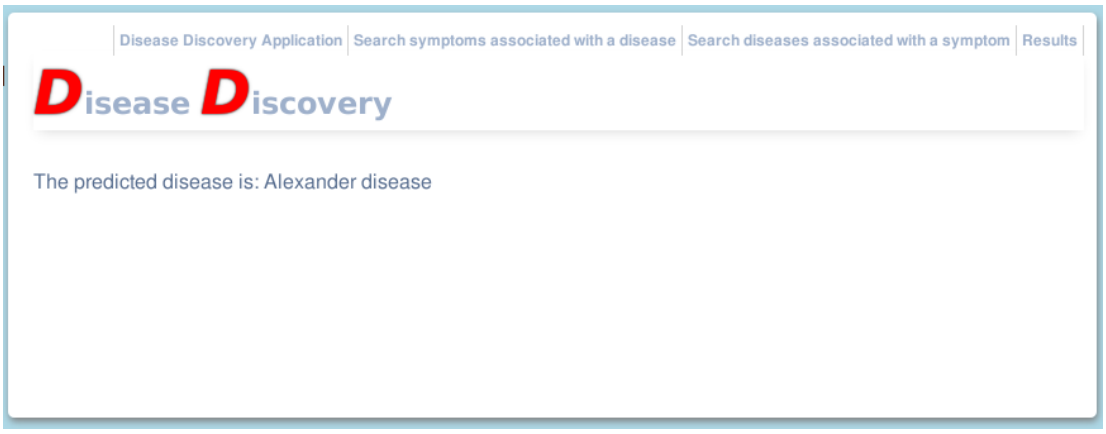


Figura 7: Resultat d'una predicció

La segona funcionalitat consisteix en donada una malaltia, mostrar tots aquells símptomes relacionats amb ella, aquestes relacions s'agafen directament de les existents a la base de dades, és a dir, no es fa cap predicció en aquest apartat ja que no té sentit. Al mateix temps pots clicar sobre un d'aquests símptomes i es fa una cerca automàtica a NIH que cerca la informació a moltes fonts mèdiques amb diverses especialitats. En la Figura 8, es pot veure un exemple de símptomes relacionats amb una malaltia, i en la Figura 9, el resultat d'una cerca a NIH d'un d'aquests símptomes.

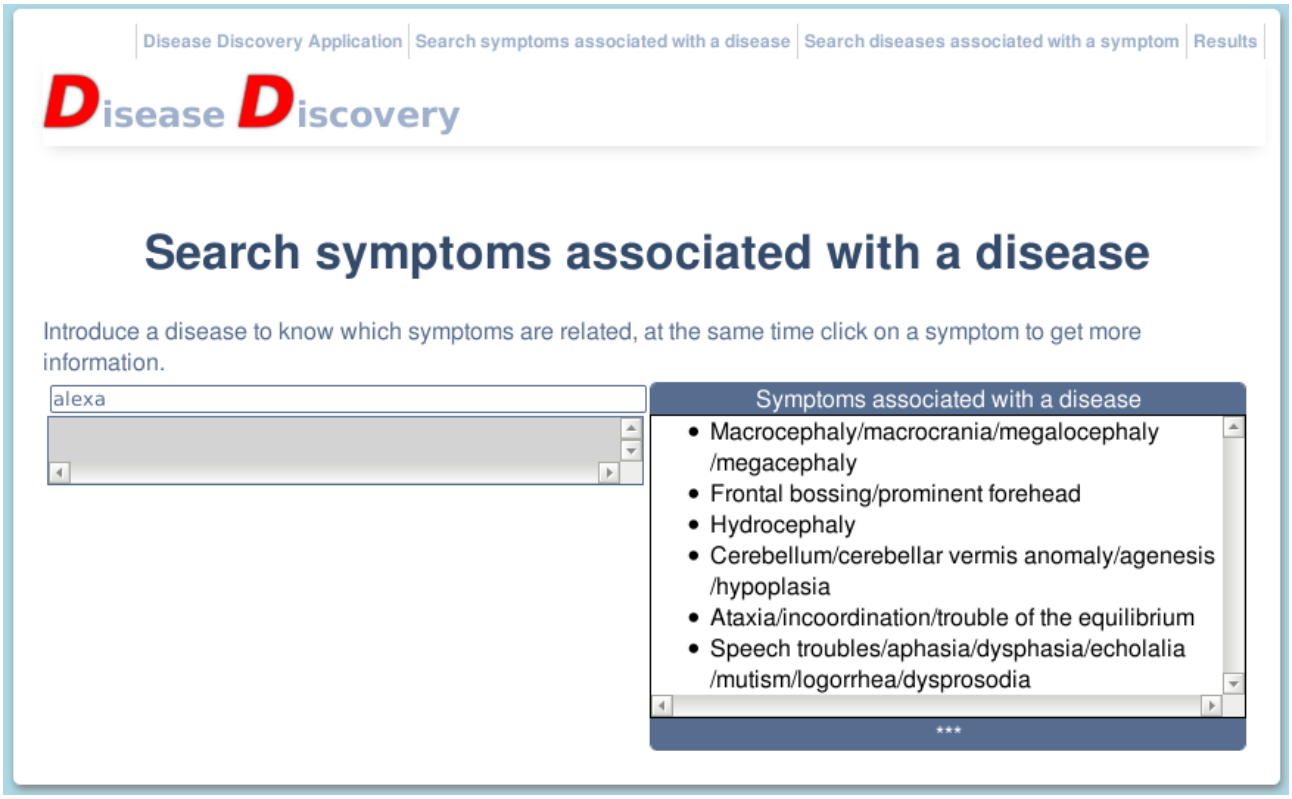


Figura 8: Exemple de símptomes relacionats amb l'Alexander Disease

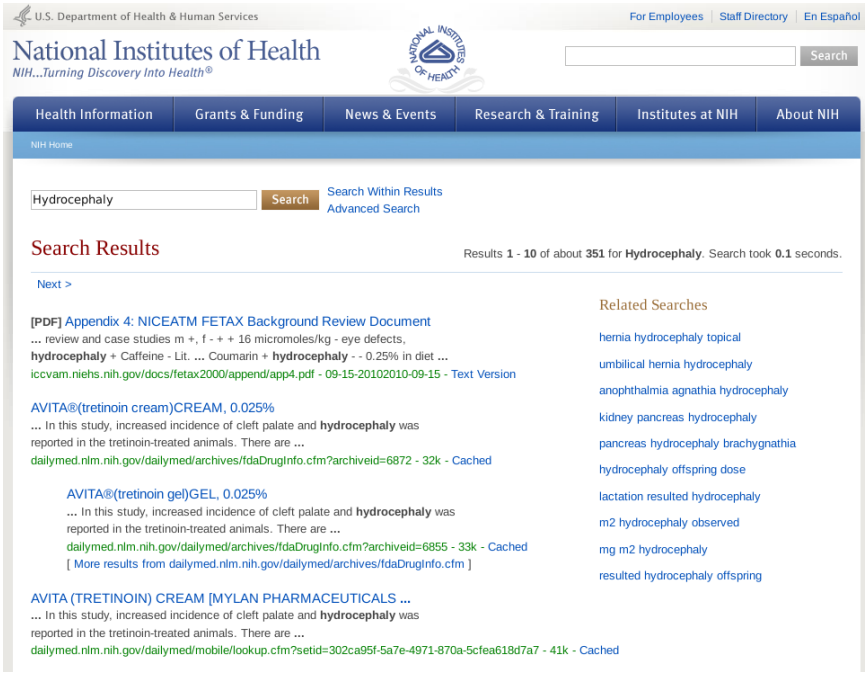


Figura 9: Cerca automàtica del símptoma Hydrocephaly a NIH

Finalment l'últim apartat ofereix la mateixa funcionalitat que l'anterior, però a la inversa, és a dir, malalties relacionades amb un símptoma, en la Figura 10 se'n pot veure un exemple.

Figura 10: Exemple de malalties relacionades amb un símptoma

4 Discussió

En aquest document, s'ha explicat tot el funcionament de l'aplicació Disease Discovery, així com el seu desenvolupament. Aquesta aplicació es capaç de fer prediccions entorn a un total de 7038 malalties i disposa d'un total de 16404 símptomes per la seua predicció. A més a més, té un total de 49981 registres que representen relacions entre símptomes i malalties.

Aquesta gran base de dades és possible gràcies que recopila informació de les bases de dades més grans i fiables a nivell mèdic del món. Al mateix temps al tenir tota aquesta informació concentrada en una sola base de dades, la converteix en un punt de referència perfecte per no tenir que consultar diverses fonts.

A més a més, el potencial recau en el fet de que no tan sols conté malalties de caràcter més general, sinó que també conta amb malalties estranyes, de les que normalment no es disposa de tanta informació i per tant diagnosticar-les pot ser un problema.

Per altra banda no cal oblidar que no existeix cap aplicació exactament com aquesta, ja que normalment no et fan una predicció donats uns símptomes, sinó que t'ofereixen informació relacionada amb una malaltia o un símptoma, per tant cobreix una necessitat que fins ara no estava coberta.

Els resultats obtinguts són força bons, ja que no hi ha massa solapament de la informació en els classificadors, i això assegura una predicció exacta de les malalties i per tant li dóna una certa fiabilitat de cara a l'ús final de l'aplicació.

5 Conclusions

L'aplicació al final ha complert el seu objectiu i és capaç de fer prediccions amb una fiabilitat força gran per ser considerada com una eina de referència per als metges, sobretot en l'àmbit de les malalties rares.

Tant mateix aquesta és una primera versió, que pot millorar moltíssim i gaudir de més robustesa, més velocitat de predicció, una base de dades molt més gran, emprar millors classificadors, etc.

Per això un primer pas seria trobar la forma de reduir l'ús de memòria dels classificadors per poder emprar algorismes millors i més ràpids, com SVM (Support Vector Machines), ja que tal i com s'ha plantejat en aquesta primera versió no és factible, ja que es requereix més de 98Gb de RAM. Per aquest treball hi ha dos possibilitats i l'opció, fins i tot, de combinar-les:

1. Fer categories de malalties, així es reduirà la mida dels dataset, i per tant les prediccions i els entrenaments necessitaran menys RAM i seran més ràpides.
2. Implementar o buscar altres plataformes de mineria de dades, que funcionin amb un llenguatge diferent de Java, ja que aquest té el problema d'emprar molta RAM, fet que no és acceptable per volums de dades molt grans com el que ens trobem.

Al classificar les malalties, en petits grups es podran generar més instàncies, això provocarà que es pugui generar més soroll (fins un cert límit) i per tant augmentar la robustesa del sistema.

Per altra banda, seria bo augmentar les fonts d'informació sobretot en l'àmbit de les malalties rares, per tant s'hauria de parlar amb diversos experts en el tema per saber d'on extreure aquesta informació.

Finalment, es pot concloure dient que és una gran idea, que cobreix una necessitat que fins ara no estava abordada, però que faria falta posar-hi aquesta serie de millores per a que pogués servir com una eina d'ús professional. Tot i això, ara per ara és capaç d'oferir prediccions força bones, amb l'única mancança que només s'ha pogut fer ús d'un classificador força simple respecte als altres.

6 Bibliografia

6.1 Fonts de consulta de caràcter general:

- http://docs.oracle.com/javase/6/docs/api/ [Documentació oficial de Java]
- http://weka.wikispaces.com/ [Wiki de Weka]
- http://docs.jquery.com/ [Documentació jQuery]
- http://tomcat.apache.org/tomcat-6.0-doc/ [Documentació general d'Apache Tomcat]
- http://ant.apache.org/manual/index.html [Documentació sistema d'automatització de tasques Apache Ant]
- http://www.viquipedia.cat [Enciclopèdia lliure]
- http://www.w3schools.com/ [Referència per la programació web]

6.2 Llibres i altres tipus de documents

- “Beggining Java Server Pages”, Wrox, ISBN 0-7645-7485-X [Introducció a la tecnologia JSP]
- “Data Mining, Practical practical machine learning tools and techniques”, Morgan Kaufmann Publishers, ISBN 0-12-088407-0 [Mineria de dades en concret amb Weka]

6.3 Fonts concretes d'informació

- http://weka.sourceforge.net/doc/weka/classifiers/trees/RandomTree.html [Doc. Random tree]
- http://weka.sourceforge.net/doc.packages/hiddenNaiveBayes/weka/classifiers/bayes/HNB.html [Doc. HNB]
- http://weka.sourceforge.net/doc/weka/classifiers/functions/SMO.html [Doc. SMO]
- http://www.kernel-machines.org/publications [Diferents fonts molt interessants sobre Support Vector Machines]
- http://analisisydecision.es/monografico-clasificacion-con-svm-en-r/ [Explicació introductòria dels SVM]

- http://weka.wikispaces.com/Use+Weka+in+your+Java+code [Emprar Weka dins de Java]
- http://planetmath.org/encyclopedia/BayesTheorem.html [Teorema de Bayes]
- http://en.wikipedia.org/wiki/Bayesian_network [Xarxes bayesianes]

6.4 Altres fonts

- wekalist@list.scms.waikato.ac.nz [Llistes de correu de Weka]
- users@tomcat.apache.org [Llistes de correu de Tomcat]